

FIG. 1

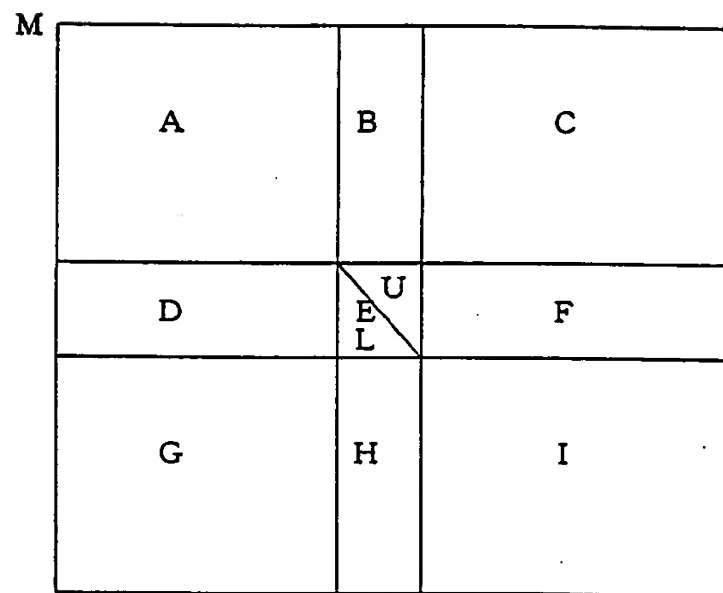


FIG. 2

E

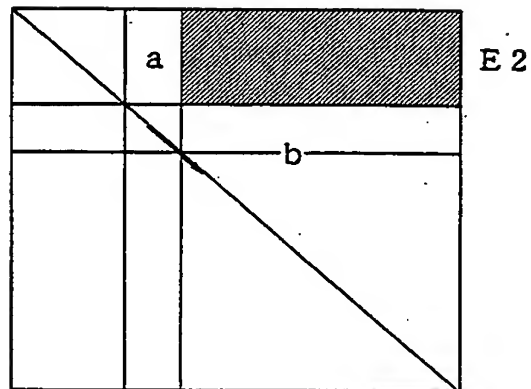


FIG. 3

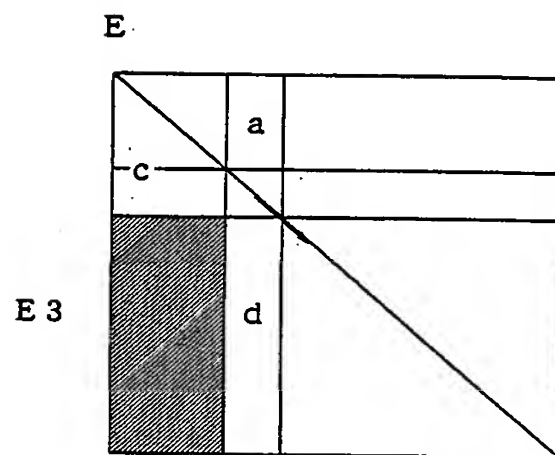


FIG. 4

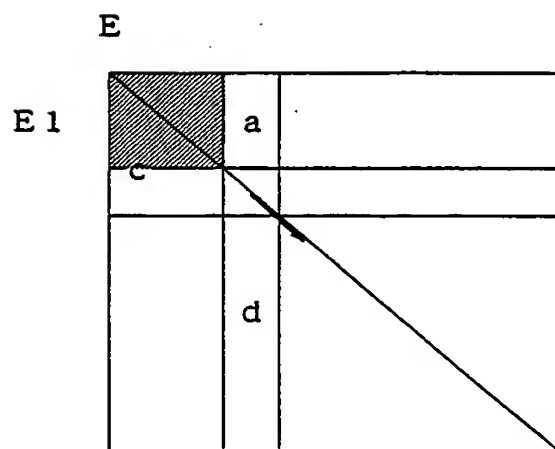


FIG. 5

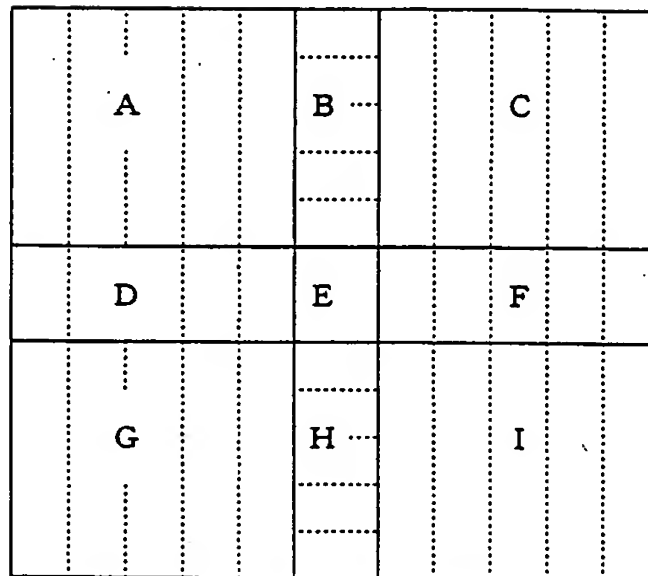


FIG. 6

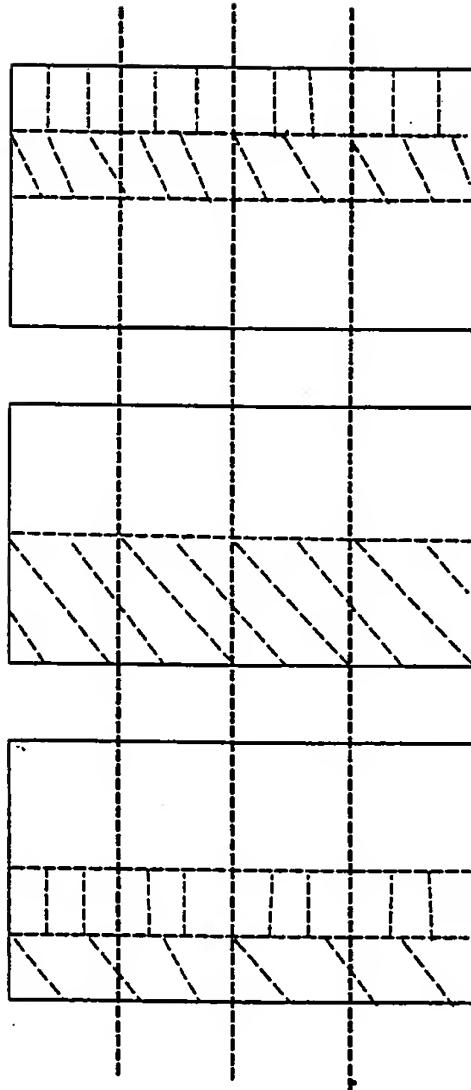


FIG. 7E

FIG. 7C

FIG. 7A

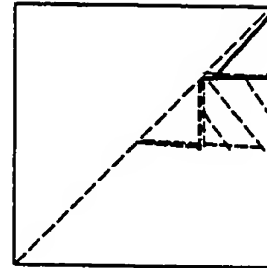


FIG. 7F

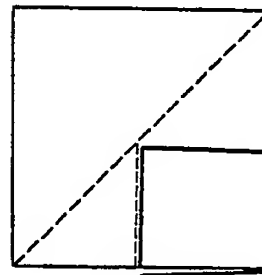


FIG. 7D

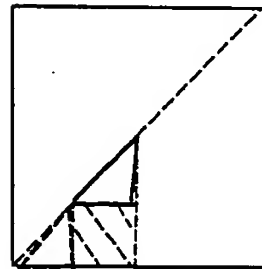
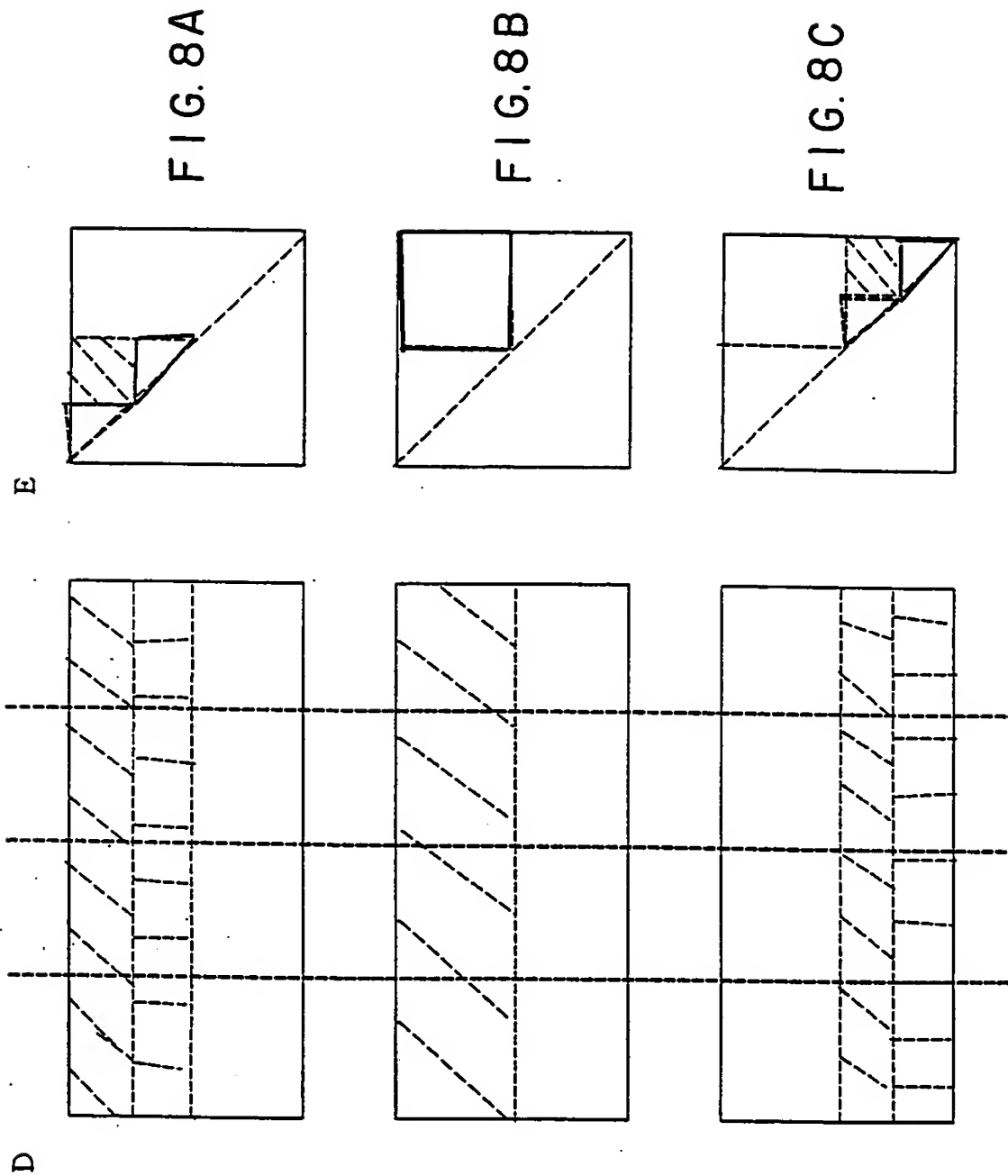


FIG. 7B

E





```
c  PARALLEL ALGORITHM OF INVERSE MATRIX
  subroutine inversion(a,k,n)
  shared array a(k,n), ip(n)
  create threads
  set nothrd and numthrd
c  nothrd IS THREAD NUMBER 1 ~ #TH, numthrd=#TH (TOTAL NUMBER OF THREADS)
  blk=BLOCK WIDTH
  nb=(n+blk-1)/blk
  do i=1,nb-1
    nbase=(nb-1)*blk
    call LU(a,k,n,nbase,blk,ip,nothrd,numthrd)
      ! LU DECOMPOSITION IS PERFORMED ON SPECIFIED BLOCK, ROW BLOCK IS
UPDATED,
      ! OPERATION FOR UPDATING LOWER RIGHT SQUARE BLOCK MATRIX ARE PERFORMED
      ! IN PARALLEL. ROW INTERCHANGING INFORMATION IS RETURNED TO
      ip(nbase+1:nbase+blk).
      ! FOR PARALLEL ALGORITHM OF THIS PORTION, REFER TO JAPANESE PATENT
      APPLICATION NO. HEI-12-358232
    do i=nbase+1,nbase+blk
      if(ip(i)>i) then
        exchange(a(i,1:nbase),a(ip(i),1:nbase))
      endif
    enddo
    call update(a,k,n,nbase,blk,nothrd,numthrd)
  enddo
  nbase=(nb-1)*blk
  blklast=n-nbase
  call LU(a,k,n,nbase,blklast,nothrd,numthrd)
  call update(a,k,n,nbase,blklast,nothrd,numthrd)
  return
end
```

F I G. 9

```

c  ROUTINE OF UPDATING REMAINING PORTION ACCORDING TO INFORMATION
    ABOUT BLOCK LU DECOMPOSITION
    subrouine update(a, k, n, nbase, blk, nothrd, numthrd)
    call b-update(a, k, n, nbase, blk, nothrd, numthrd)
    call d-update(a, k, n, nbase, blk, nothrd, numthrd)
    call c-update(a, k, n, nbase, blk, nothrd, numthrd)
    call a-update(a, k, n, nbase, blk, nothrd, numthrd)
    call g-update(a, k, n, nbase, blk, nothrd, numthrd)
    BARRIER SYNC
    if(nothread=1) then
c  UPDATE 1 OF e
    call e-update1(a(nbase+1, nbase+1), k, n, blk)
    endif
    BARRIER SYNC
    len←(nbase+numthrd-1)/numthrd
    is←(nothrd-1)*len+1
    ie←nothrd*len
    call df-update(a(nbase+1, 1), k, n, a(nbase+1, nbase+1), is, ie, blk)
    nbase2←nbase+blk
    len←(n-nbase2+numthrd-1)/numthrd
    is2←nbase2+(nothrd-1)*len+1
    ie2←nbase2+nothrd*len
    call df-update(a(nbase+1, 1), k, n, a(nbase+1, nbase+1), is, ie, blk)
    BARRIER SYNC
    if(nothread=1) then
c  UPDATE 2 OF e
    call e-update2(a(nbase+1, nbase+1), k, n, blk)
    endif
    BARRIER SYNC
    len←(nbase+numthrd-1)/numthrd
    is←(nothrd-1)*len+1
    ie←nothrd*len
    call bh-update(a(1, nbase+1), k, n, a(nbase+1, nbase+1), is, ie, blk)
    nbase2←nbase+blk
    len←(n-nbase2+numthrd-1)/numthrd
    is2←nbase2+(nothrd-1)*len+1
    ie2←nbase2+nothrd*len
    call bh-update(a(1, nbase+1), k, n, a(nbase+1, nbase+1), is, ie, blk)
    BARRIER SYNC
    if(nothread=1) then
c  UPDATE 3 OF e
    call e-update3(a(nbase+1, nbase+1), k, n, blk)
    endif
    BARRIER SYNC
    len=(n+numthrd-1)/numthrd
    is=(nothrd-1)*len+1
    ie=min(n, nothrd*len)
    if(ip(i)>i) then
    exchange(a(is:ie, i), a(is:ie, ip(i)))
    endif
    enddo
    BARRIER SYNC

    eliminate threads

    return
end

```

FIG. 10

```

c  UPDATE OF BLOCK B
   subroutine b-update(a, k, n, nbase, blk, nothrd, numthrd)
   shared array a(k, n)
   len←(nbase+numthrd-1)/numthrd
   is1←(nothrd-1)*len
   ie1←nothrd*len
   a(is:ie, iof+1:iof+blk)
   ←a(is:ie, iof+1:iof+blk)*TRU-U(iof+1:iof+blk, iof+1:iof+blk)-1
   ! TRU-U UPPER TRIANGULAR MATRIX WITH DIAGONAL ELEMENT=1.0
   return
   end

c  UPDATE OF BLOCK D
   subroutine d-update(a, k, n, nbase, blk, nothrd, numthrd)
   shared array a(k, n)
   len←(nbase+numthrd-1)/numthrd
   is1←(nothrd-1)*len
   ie1←nothrd*len
   iof=nbase
   a(is:ie, iof+1:iof+blk)
   ←TRL(iof+1:iof+blk, iof+1:iof+blk)-1*a(is:ie, iof+1:iof+blk)
   ! TRL INDICATES LOWER TRIANGULAR MATRIX
   return
   end

```

F I G. 1 1

```

c  UPDATE OF BLOCK C
    subroutine c-update(a, k, n, nbase, blk, nothrd, numthrd)
    shared array a(k,n)
    nbase2←nbase+blk
    iof=nbase
    len←(n-nbase2+numthrd-1)/numthrd
    is2←nbase2+(nothrd-1)*len+1
    ie2←nbase2+nothrd*len
    a(1:iof, is2: ie2)
    ←a(1:iof, is2: ie2)
    -a(1:iof, iof+1:iof+blk)*a(iof+1:iof+blk, is2: ie2)
    return
    end

c  UPDATE OF BLOCK A
    subroutine a-update(a, k, n, nbase, blk, nothrd, numthrd)
    shared array a(k,n)
    len←(nbase+numthrd-1)/numthrd
    is2←(nothrd-1)*len
    ie2←nothrd*len
    iof=nbase
    a(1:iof, is2: ie2)
    ←a(1:iof, is2: ie2)
    -a(1:iof, iof+1:iof+blk)*a(iof+1:iof+blk, is2: ie2)
    return
    end

```

F I G. 1 2

```

c  UPDATE OF BLOCK G
   subroutine g-update(a, k, n, nbase, blk, nothrd, numthrd)
   shared array a(k, n)
   len ← (nbase+numthrd-1)/numthrd
   is2 ← (nothrd-1)*len
   ie2 ← nothrd*len
   iof ← nbase+blk
   a(iof+1:n, is2:ie2)
   ← a(iof+1:n, is2:ie2)
   -a(iof+1:n, iof+1:iof+blk)*a(iof+1:iof+blk, is2:ie2)
   return
   end

c  FIRST UPDATE OF BLOCK E
   subroutine e-update1(s, k, n, blk)
   shared array s(k, n)

   do i=1, blk
   s(1:i-1, i+1:blk) ← s(1:i-1, i+1:blk) - s(1:i-1, i)*s(i, i+1:blk)
   enddo
   return
   end

c  SECOND UPDATE OF BLOCK E
   subroutine e-update2(s, k, n, blk)
   shared array s(k, n)

   do i=1, blk
   tmp ← 1.0/s(i, i)
   s(i, 1:i-1) ← tmp*s(i, 1:i-1)
   s(i+1:blk, 1:i-1) ← s(i+1:blk, 1:i-1) - s(i, 1:i-1)*s(i+1:blk, i)
   s(i+1:blk, i) ← s(i+1:blk, i)*tmp
   s(i, i) ← tmp
   enddo
   return
   end

```

F I G. 1 3

```

c  FINAL UPDATE OF BLOCK E
   subroutine e-update3(s,k,n,blk)
   shared array s(k,n)

   do i=1,blk
     s(1:i-1,1:i-1)←s(1:i-1,1:i-1)-s(1:i-1,i)*s(i,1:i-1)
     s(1:i-1,i)←-s(1:i-1,i)*a(i,i)
   enddo
   return
end

c  UPDATE OF BLOCKS D AND F
   subroutine df-update(a,k,n,s,is,ie,len)
   shared array a(k,*),s(k,*)
   if(len<10)then
     do i=1,len
       a(1:i-1,is:ie)←a(1:i-1,is:ie)-s(1:i-1,i)*a(i,is:ie)
     enddo
   else
     if(len>=32 or len<=20)then
       len1←len/2
       len2←len-len1
     else
       len1←len/3
       len2←len-len1
     endif
     call df-update(a,k,n,s,is,ie,len1)
     a(1:len1,is:ie)
     ←a(1:len1,is:ie)-s(1:len1,len1+1:len)*a(len1+1:len,is:ie)
     call df-update(a(len1+1,1),k,n,s(len1+1,len1+1),is,ie,len2)
   endif
   return
end

```

F I G. 1 4

```
c  UPDATE OF BLOCKS B AND H
    subroutine bh-update(a, k, n, s, is, ie, len)
    shared array a(k, *), s(k, *)
    if(len<10) then
    do i=1, len
    a(is:ie, i) ← a(is:ie, i)*s(i, i)
    a(is:ie, i) ← a(is:ie, i)-a(is:ie, i+1:len)*s(i+1:len, i)
    enddo
    else
    if(len>=32 or len<=20) then
    len1 ← len/2
    len2 ← len-len1
    else
    len1 ← len/3
    len2 ← len-len1
    endif
    call bh-update(a, k, n, s, is, ie, len1)
    a(is:ie, 1:len1) ← a(is:ie, 1:len1)
                     -a(is:ie, len1+1:len)*s(len1+1:len, 1:len1)
    call bh-update(a(1, len1+1), k, n, s(len1+1, len1+1), is, ie, len2)
    endif
    return
    end
```

F I G. 15

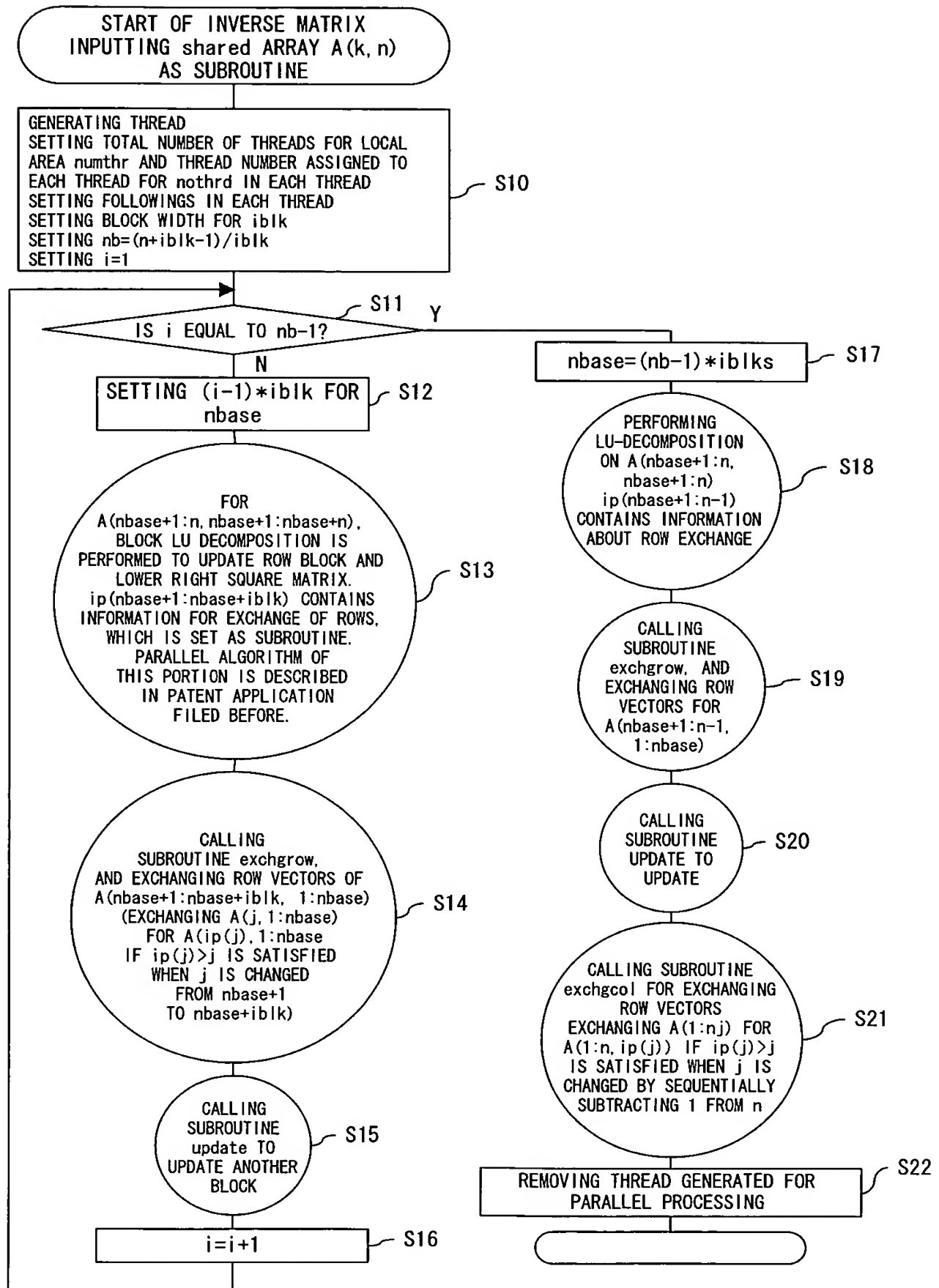


FIG. 16



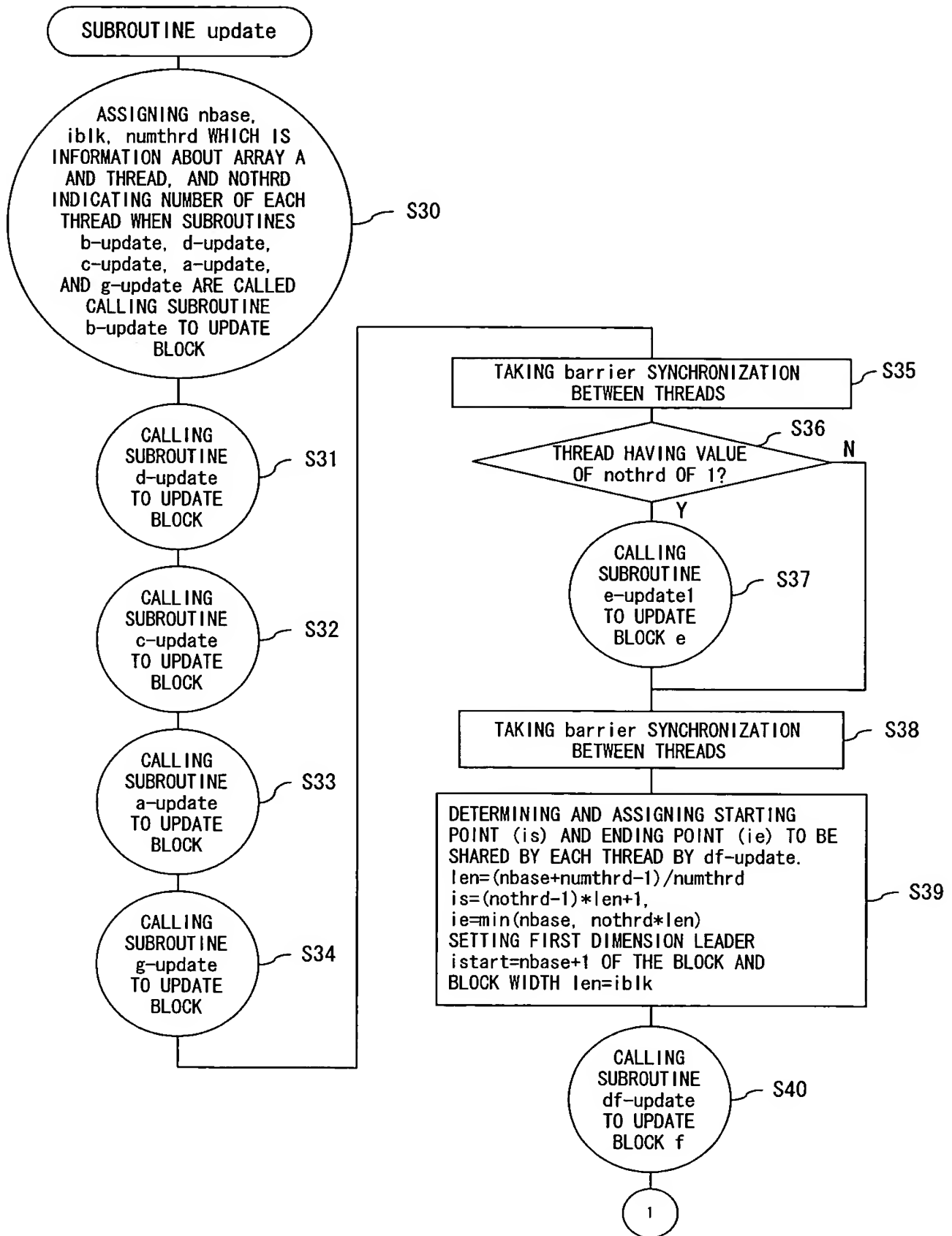


FIG. 17

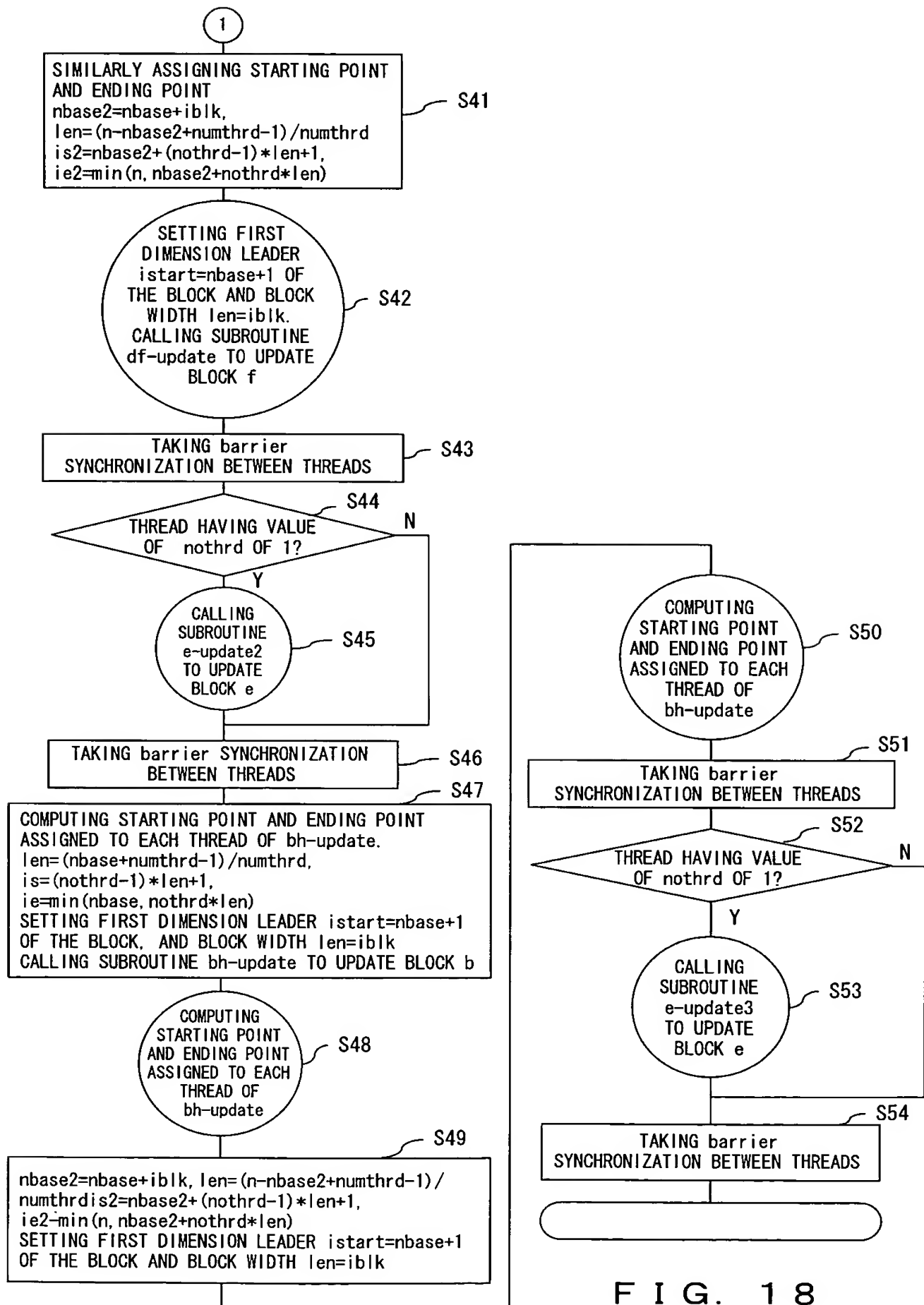


FIG. 18

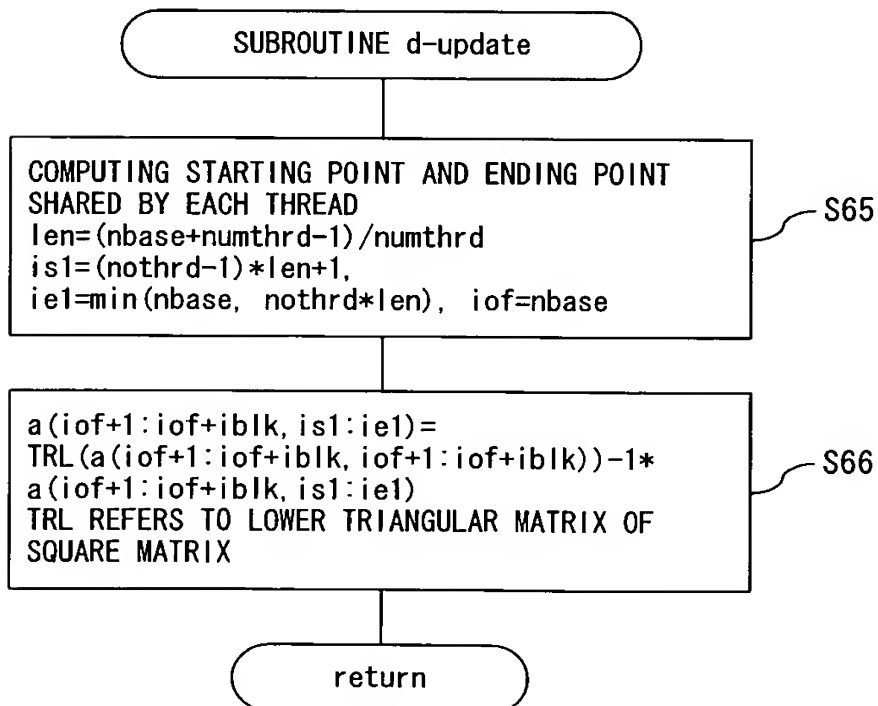
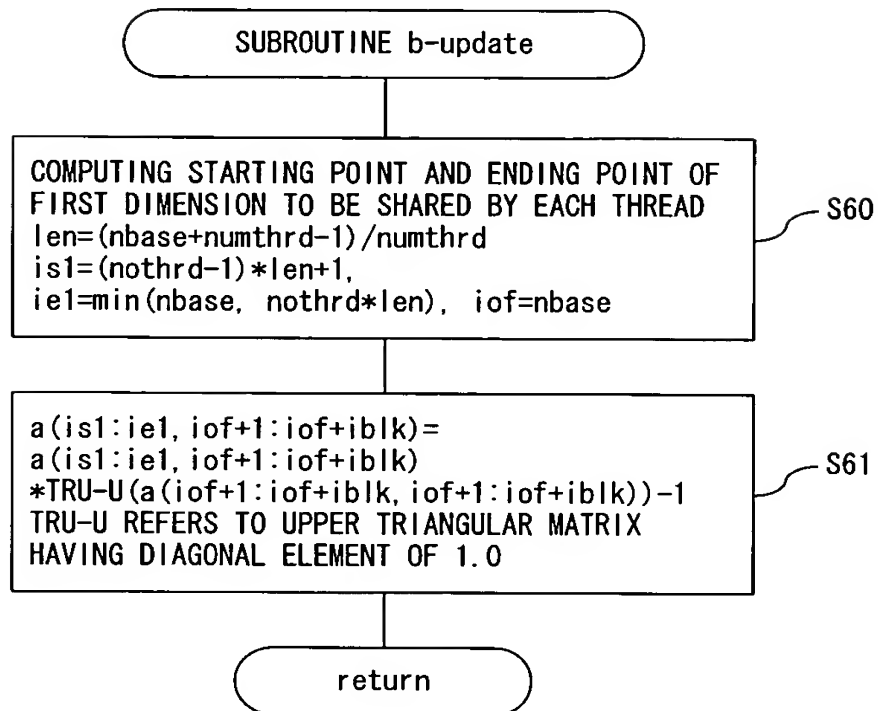


FIG. 19

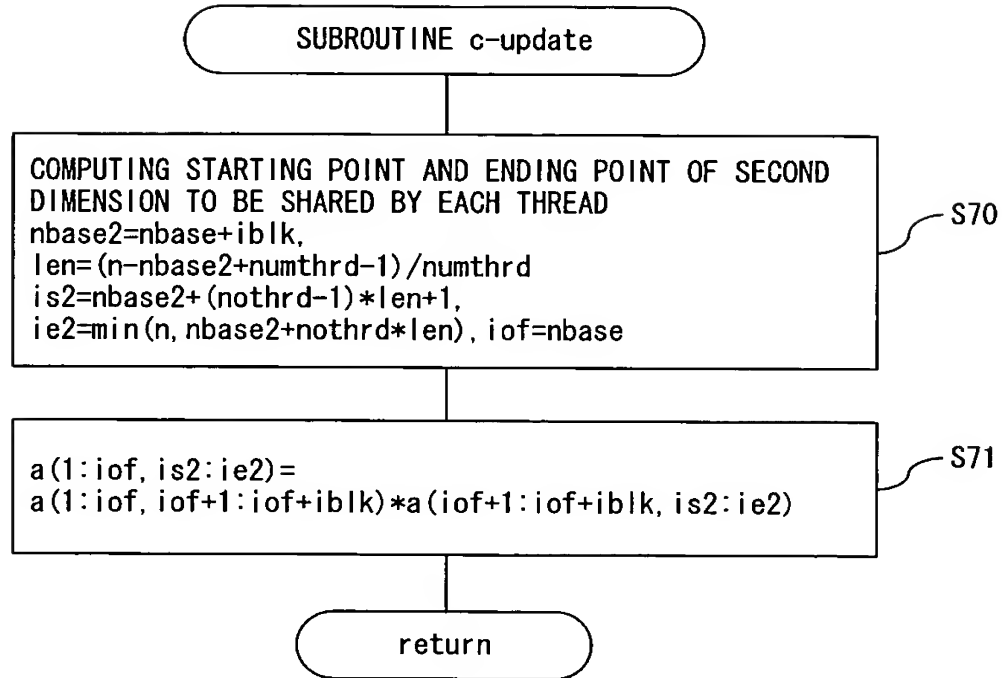


FIG. 20

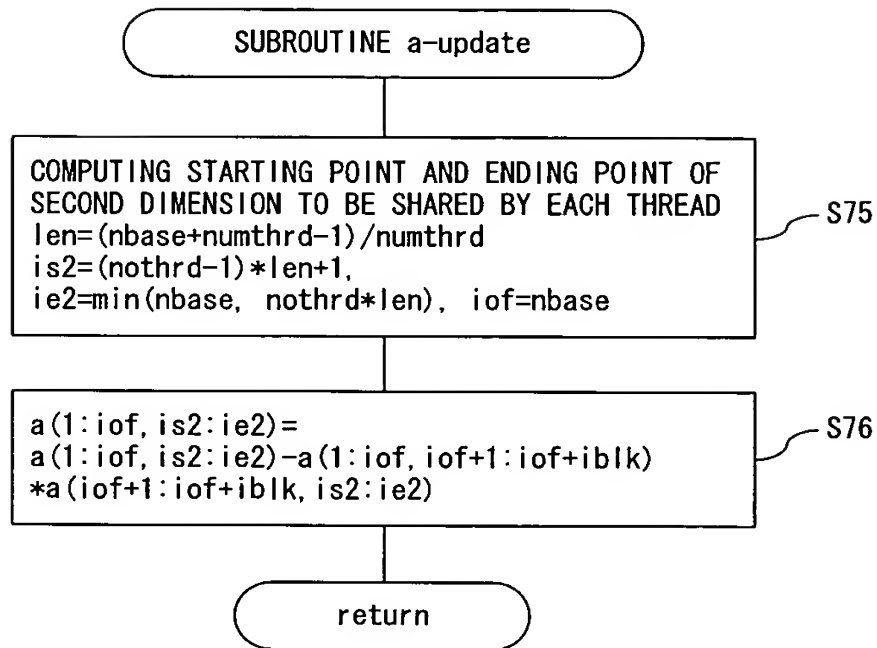
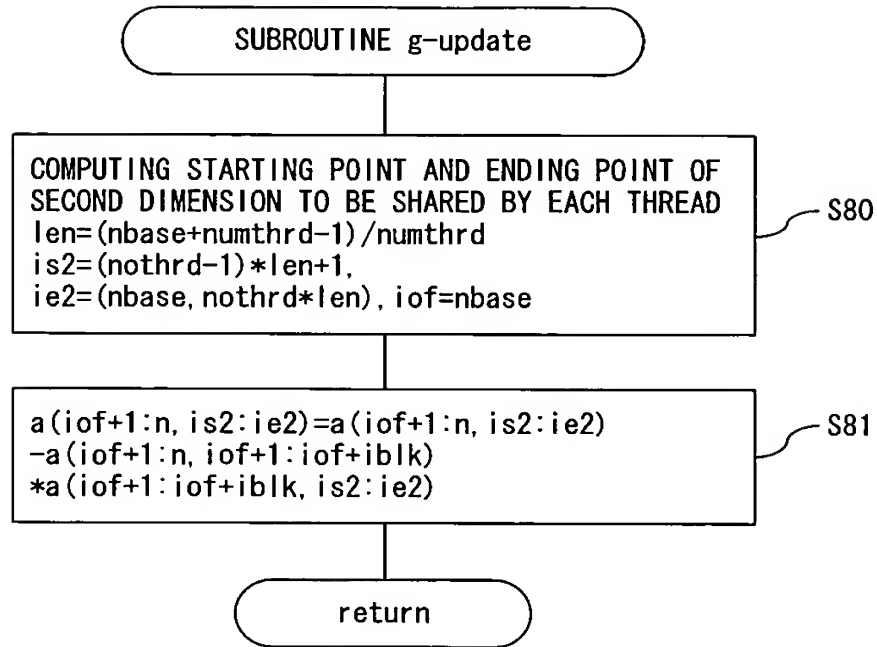


FIG. 21



F I G. 2 2

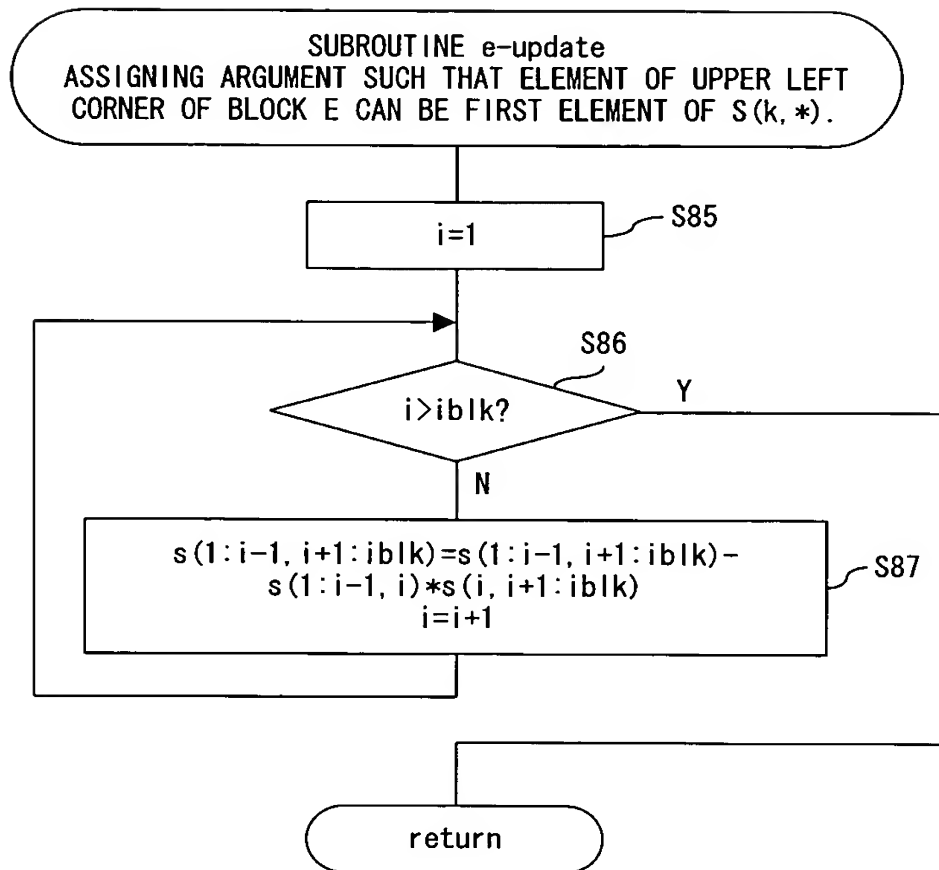
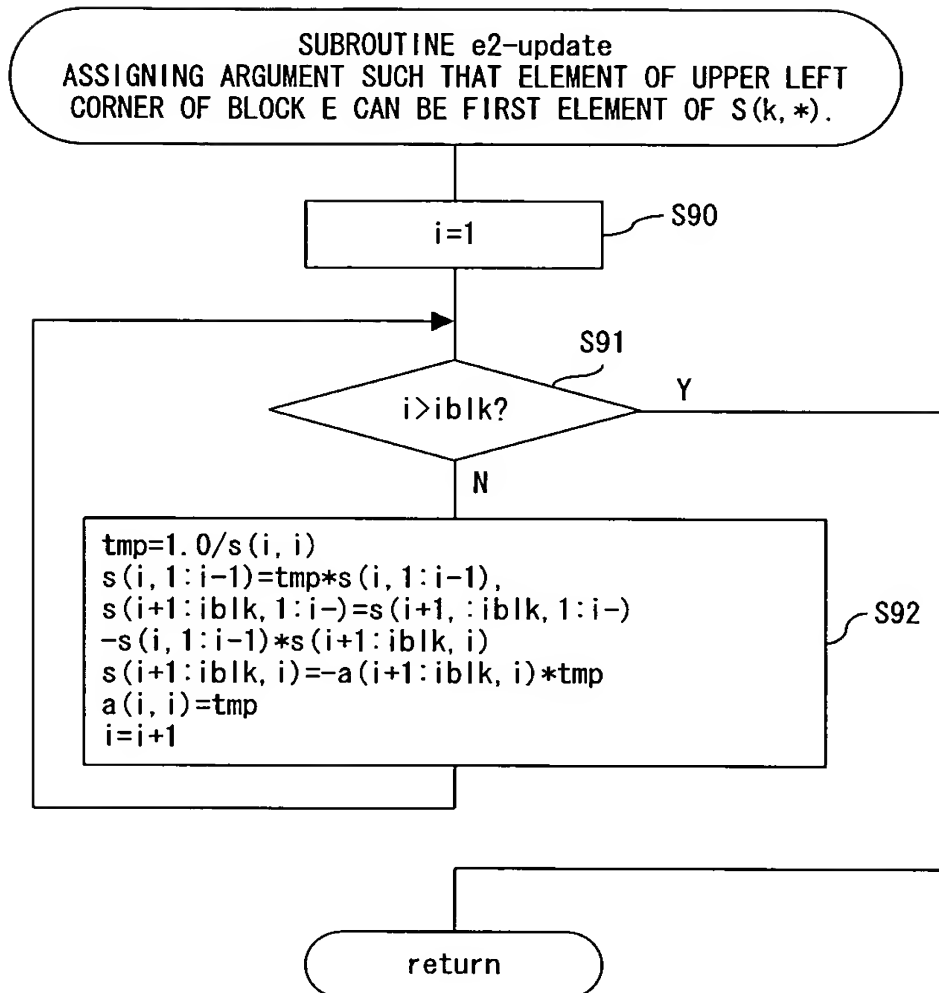


FIG. 23



F I G. 24



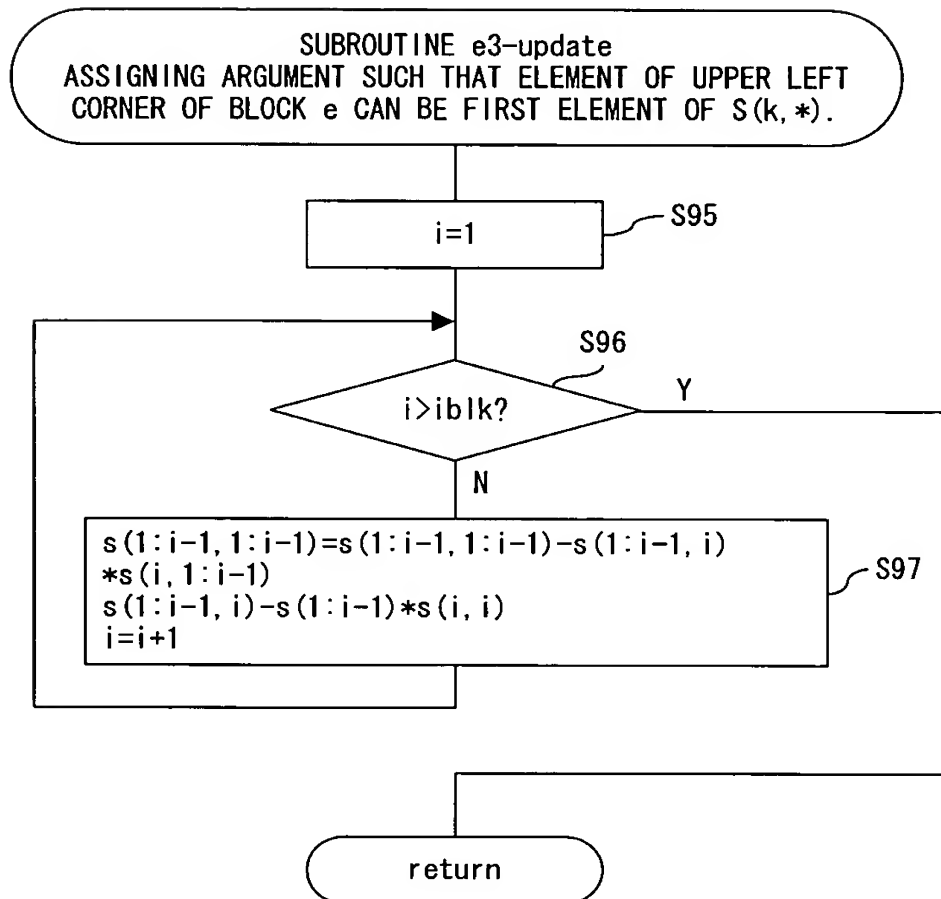


FIG. 25

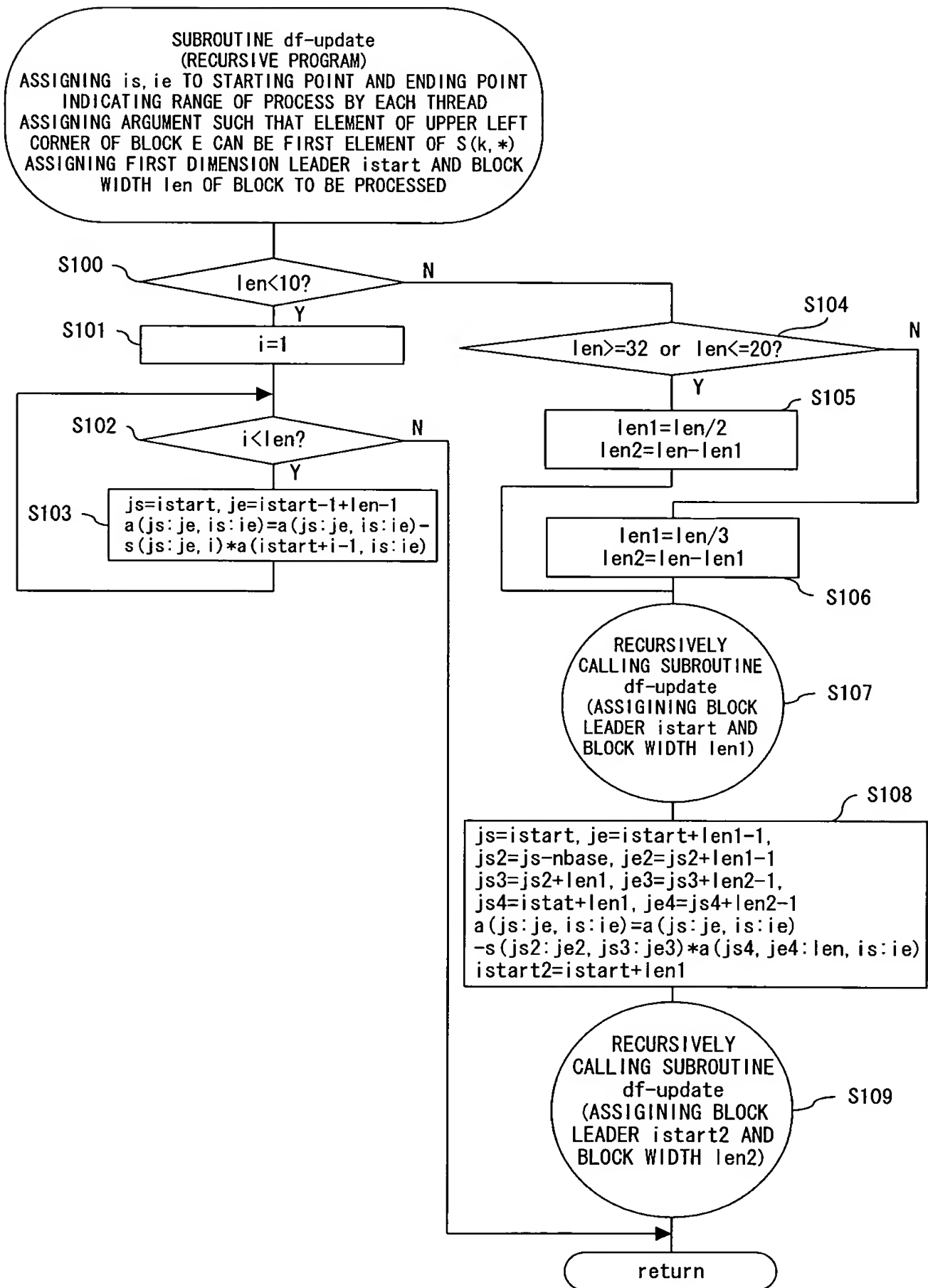


FIG. 26

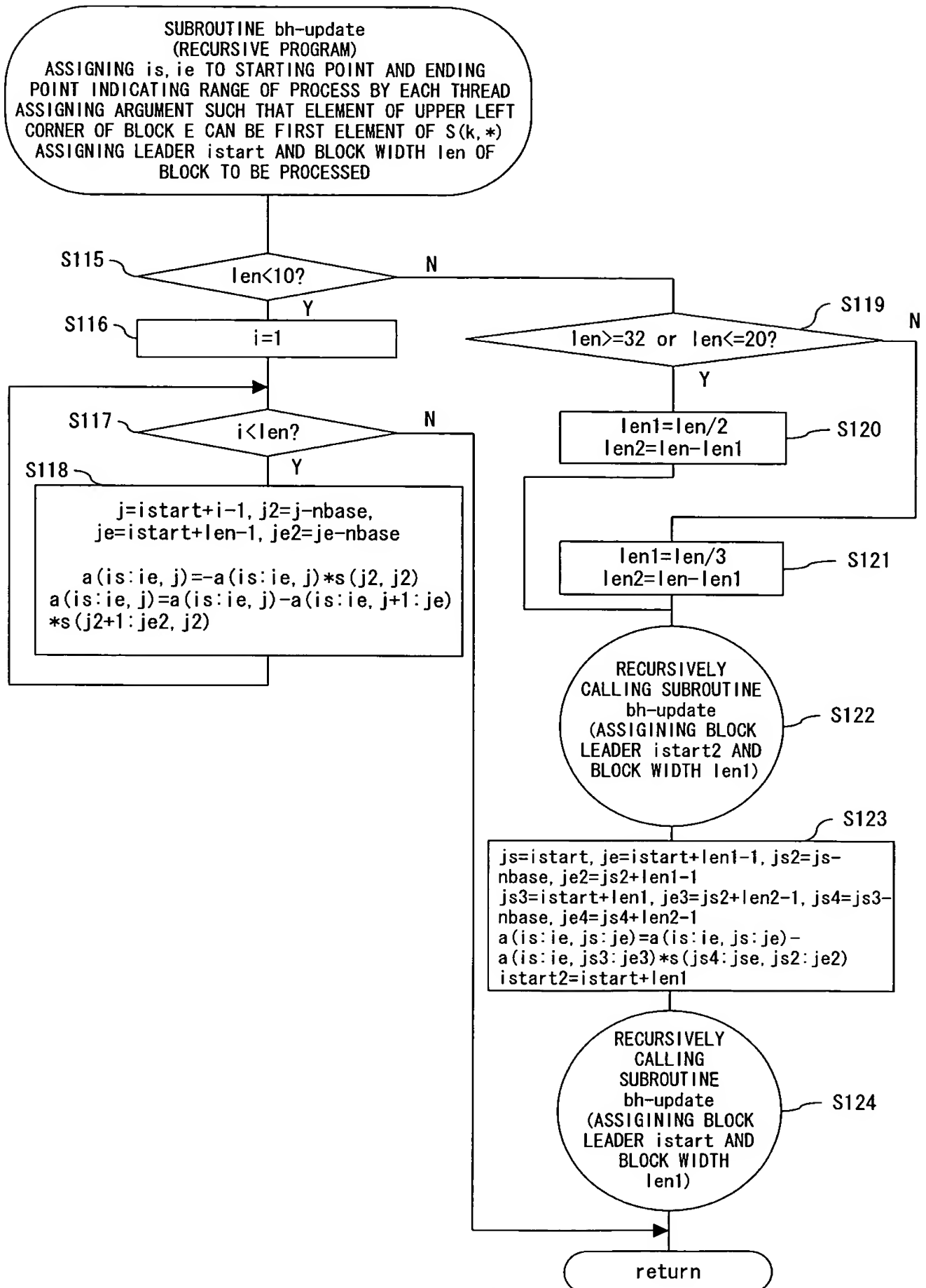


FIG. 27

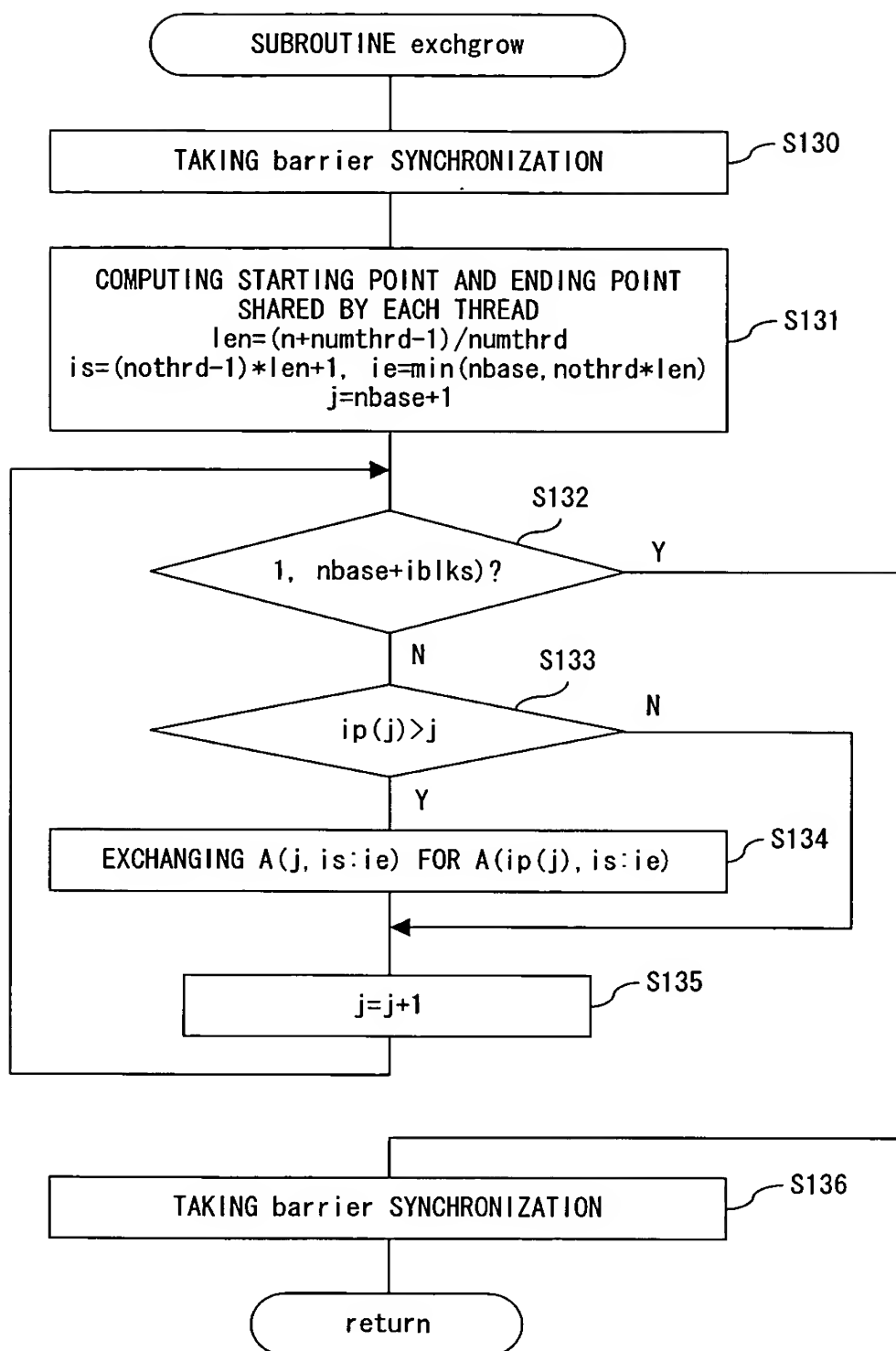


FIG. 28

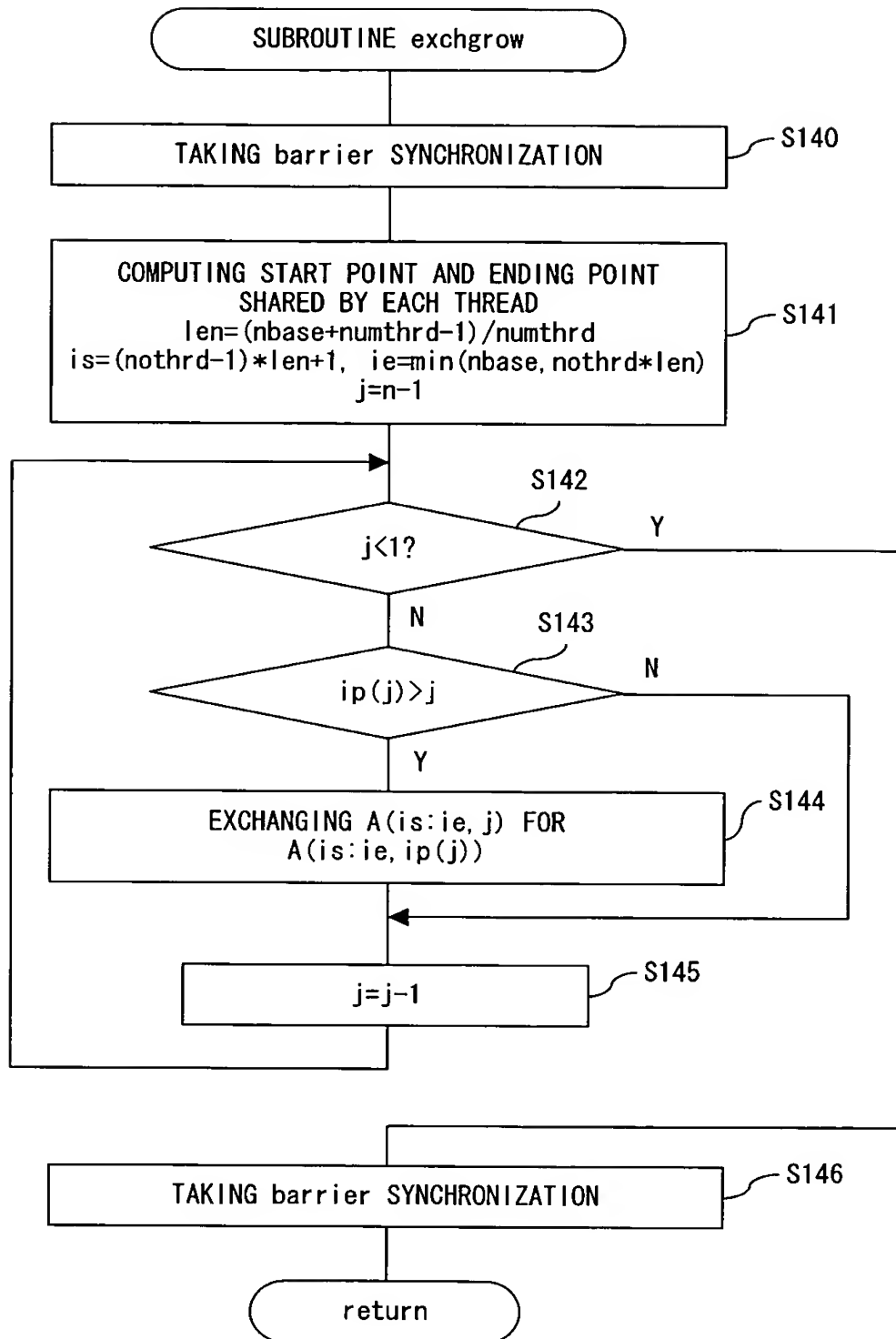


FIG. 29